

آکادمی رباتیک

دوره Advanced Tensorflow

جلسه اول : خدای برنامه نویسی (شی گرای در پایتون)



حامد قاسمی

قاری الحسین ارشد هوش مصنوعی دانشگاه تهران
{ استاد یاسین دینار
لے دسترا



نحوه استفاده شما از دوره:

✓ چند و در چند روز به تدریج به دقت یاد بگیرید؟

کلاس باید بتوانم به روشی که شما

۲۰ درصد ویدیو

۲۰ درصد مرور شما

۶۰ درصد حل تمرینات و فعالیت در گروه

ایه حرفای به مناسبت

~~۹۰٪~~

~~۴۰٪~~
۲۰٪

به اصل ماجرا

خطای متداول :

من روم همیشه توی گروه سوال بپرسم !

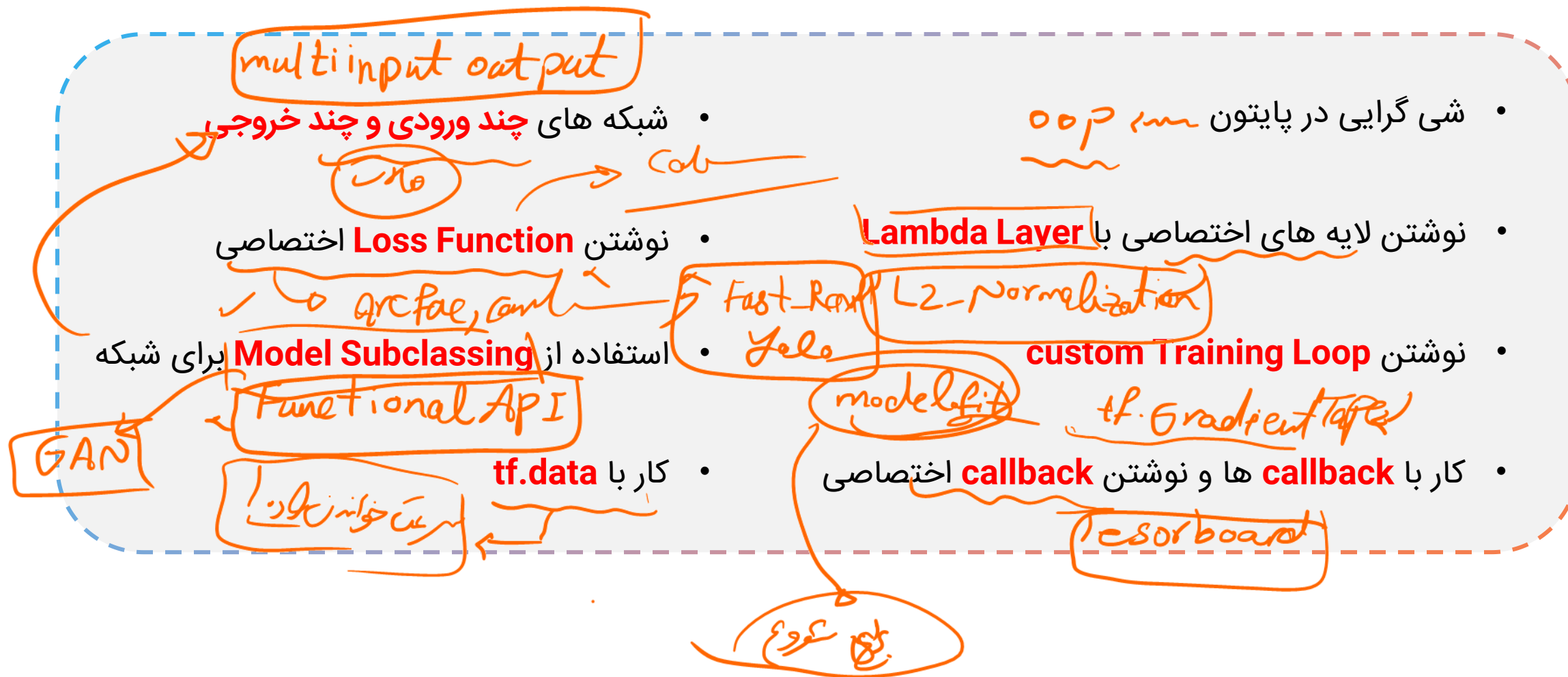
سوالاش خرابه

حرکت خوب :

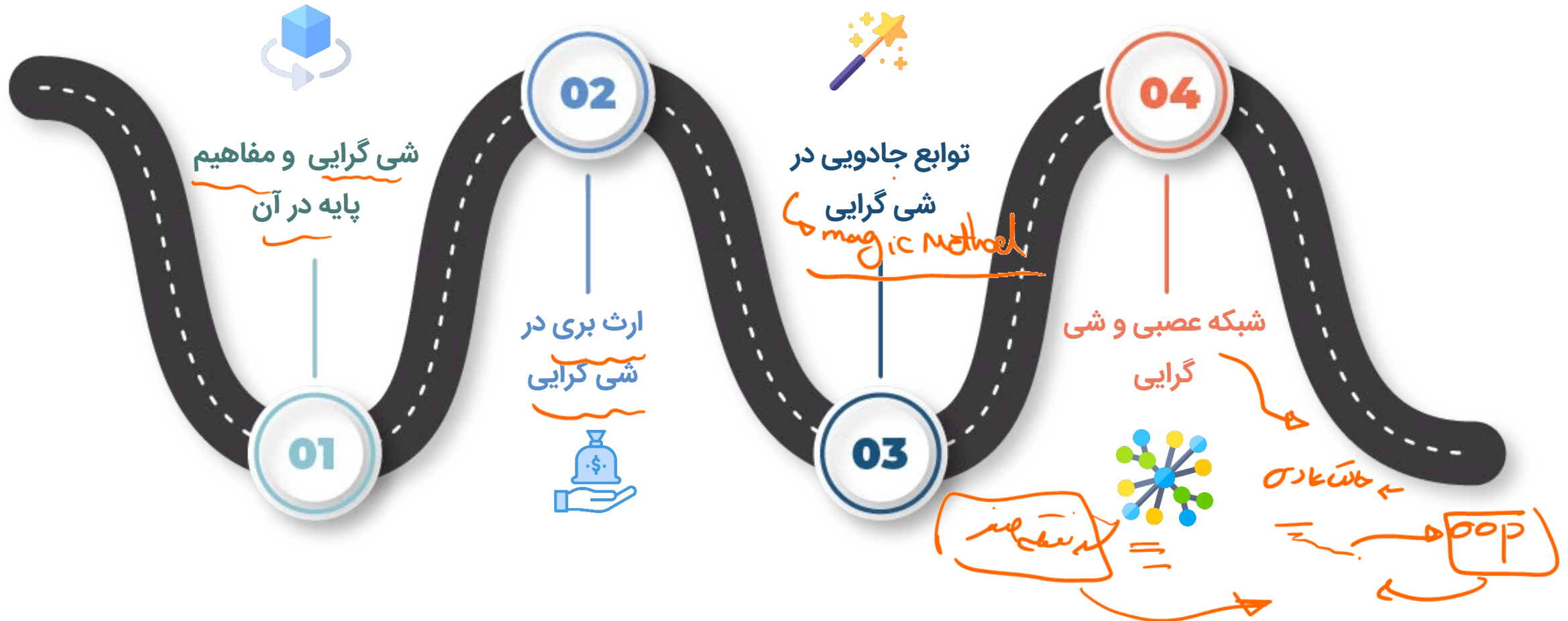
برنامه زمانی معین داشته باشید.

۱- ساعت شش و نیم
۲- صبح ۵

آنچه در این دوره خواهیم گفت:



آنچه امروز خواهیم گفت :




• فرض کنید خدا بودید و می خواستید بنده های خود را خلق کنید.


قد - وزن -
هر بنده رو جدا گانه با ویژگی ها و عملکردهای
خاص خودش می ساختم.

7 طیاره دار بار
1

موسا
مرد در سیمین



یک الگوی کلی ایجاد می کردم و از روی اون
الگو بنده ها رو می ساختم!



2

در شی گزایی همه چیز جان پیدا می کنند و می توانید به هر بخش برنامه جان بدهید

Deep

Fit

و اما



حافظه

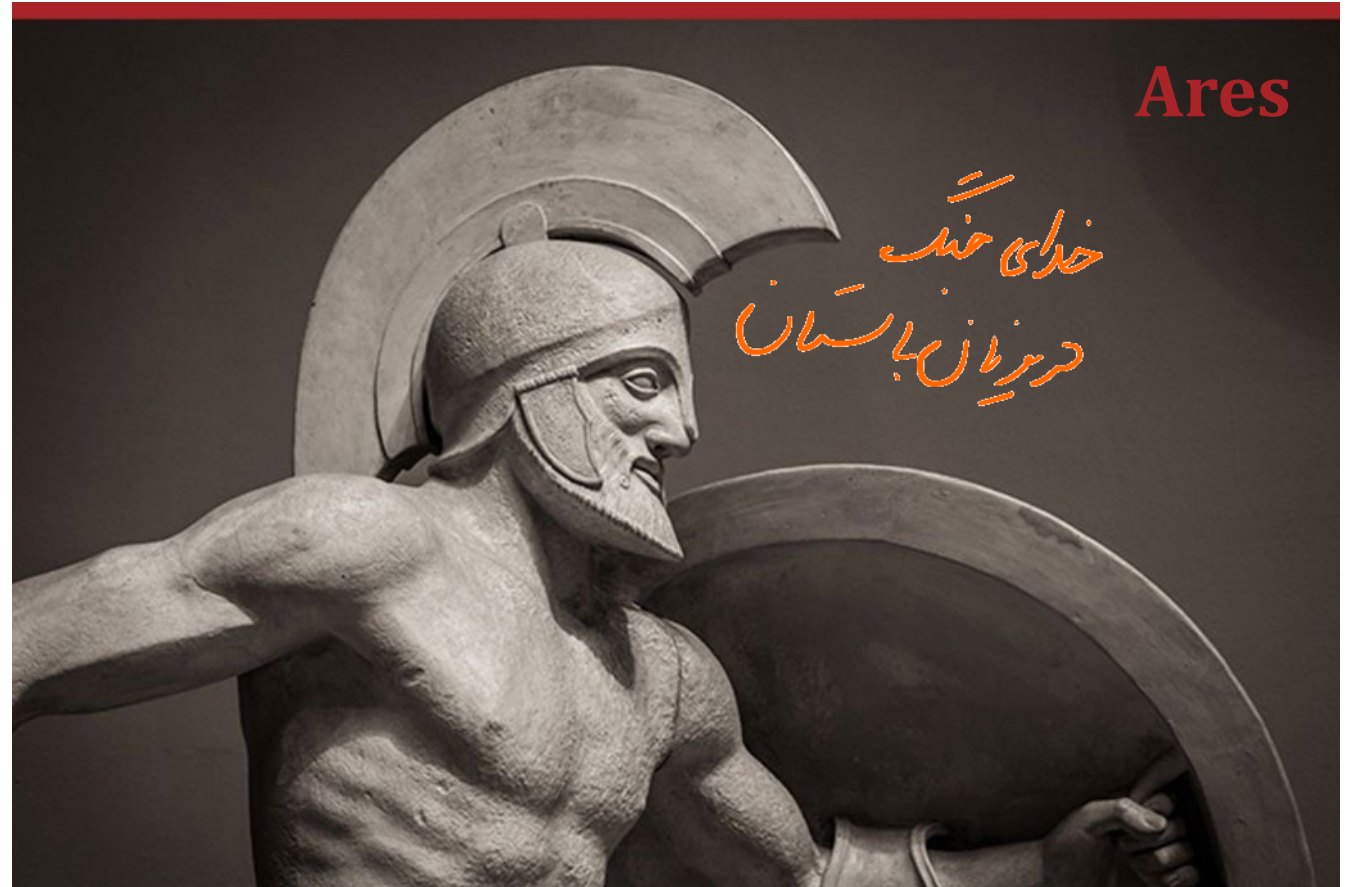
آنچه در حافظه

عسلر

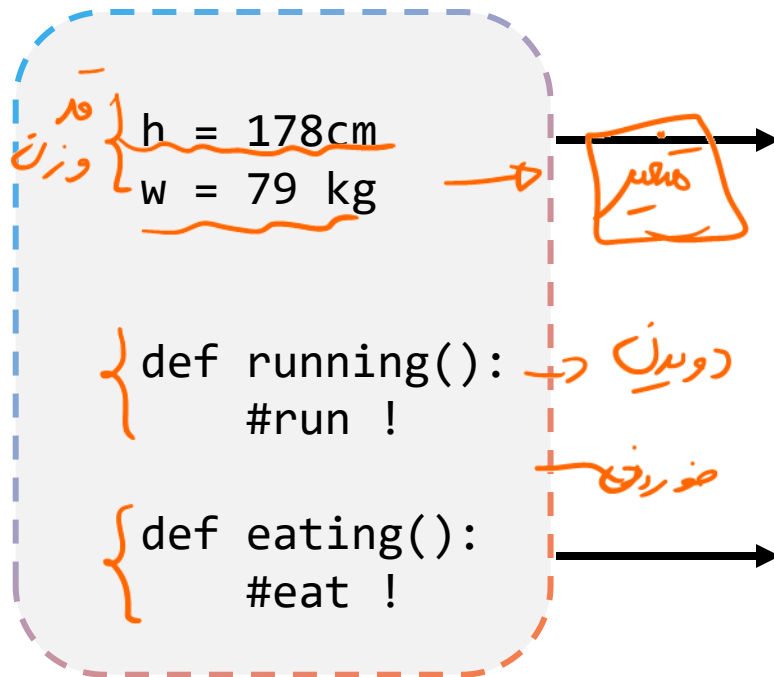
زنجیره خوانه

حتما بايد شي گرايي بلد باشيم ؟

پاشا نوحه : بله
خبر
آه درمنايه 99.9٪
عقرب سوار به سائله ستر

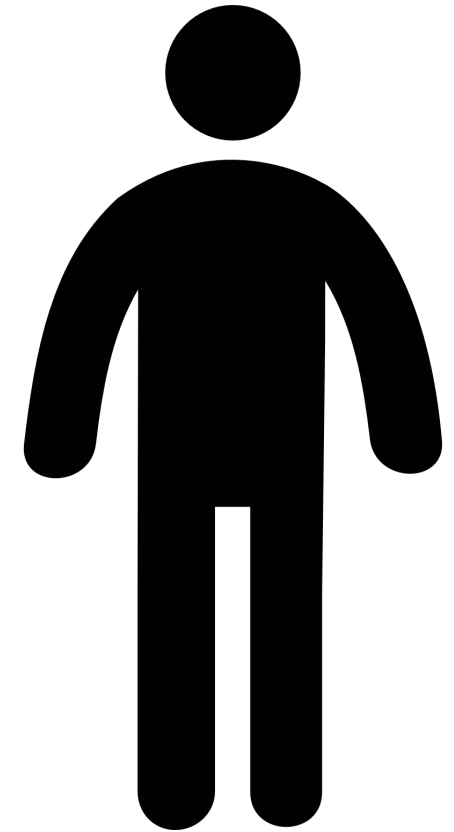


حال فرض کنید می خواهیم برنامه یک انسان را بنویسیم : (روش قدیمی)



ویژگی ها و مشخصات

کارایی که میتونه انجام بده



تعريف الكوي انسان به كمك Zeus (خدای خدایان)

```
class Human():  
    def specialMethod():  
        h = 178  
        w = 79  
    def running():  
        # run !  
  
human_1 = Human()
```

human_2 = Human() → Instance

→ Attribute

→ Method

لکه همان توابع موجود در کلاس

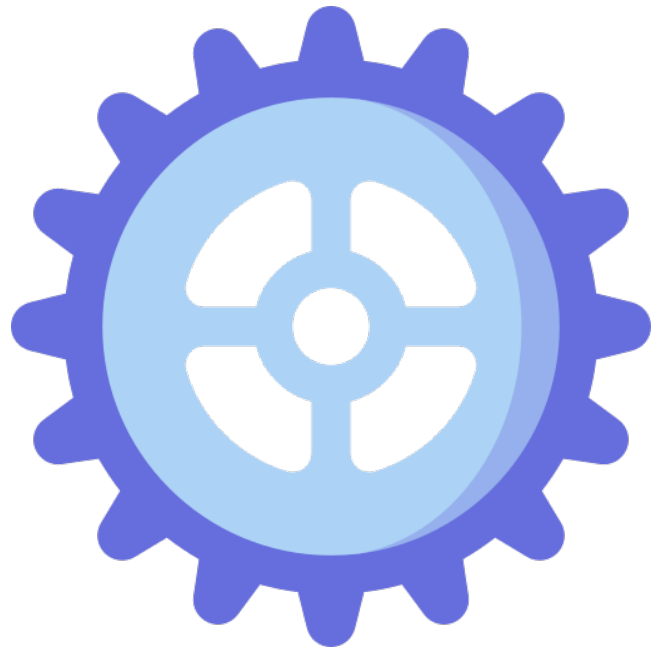
178
79

انصافات



نکته کلیدی ۱:

هر موقع شما یک **Instance** از یک کلاس تعریف می کنید، تابع Special شما به صورت خودکار اجرا می گردد.



نکته کلیدی ۲:

تمامی متدهای موجود در کلاس به صورت پیش فرض اولین آرگومان آنها **self** است. **self** همان **instance** تعریف شده است.

```
class Human():
```

__init__

```
def specialMethod(self):
```

```
self.h = 178
```

```
self.w = 79
```

```
def running(self):
```

```
pass
```

```
human_1 = Human(178, 79)
```

```
human_2 = Human(202, 101)
```

(self, h, w)

نکته کلیدی ۳:

این تابع ویژه ما یک نام خاص نیز دارد و آن نام **__init__** (بخوانید داندر init) است. به این تابع **constructor** نیز می گویند.

ما زنده



به کمک تابع **constructor** می توانیم **attribute** های کلاس خود را تنظیم کنیم.

سودمند

بریم وارد بخش کد بشیم.



می توانیم Attribute ها را به عنوان ورودی به Constructor پاس دهیم.

```
class Human():  
  
    def __init__(self, height, weight):  
        self.h = height  
        self.w = weight  
  
    def running(self):  
        pass  
  
human_1 = Human(178, 79)  
human_2 = Human(158, 61)
```



نحوه استفاده از Method

```
class Human():
```

```
    def __init__(self, height, weight, first, last):
        self.h = height
        self.w = weight
        self.first = first
        self.last = last

    def introduce(self):
        print("I am {} {}".format(self.first, self.last))
```

ویرایشگر

تعریف

```
human_1 = Human(178, 79, "hamed", "ghasemi")
```

```
human_2 = Human(158, 61, "amin", "bemani")
```

```
human_1.introduce()
```

بیا **self** رو پاس ندیم !

Instance Variable VS Class Variable

Instance

→ متغیر در سطح

متغیر کلاس

→ متغیر کلاس instance

مثال

```
class Human():  
  
    raise_parameter = 1.05  
  
    def __init__(self, height, weight, first, last):  
        self.h = height  
        self.w = weight  
        self.first = first  
        self.last = last  
  
    def introduce(self):  
        print("I am {} {}".format(self.first, self.last))  
  
    def overweight(self):  
        self.w = self.w * self.raise_parameter
```

مثالی دیگر

```
class Human():  
  
    raise_parameter = 1.05  
    num_of_instance = 0  
  
    def __init__(self, height, weight, first, last):  
        self.h = height  
        self.w = weight  
        self.first = first  
        self.last = last  
        Human.num_of_instance += 1
```

Method

(self)

Regular Method Vs Class Method Vs Static Method

← منسج (ک)

Class Method

self (cls, ...)

@classmethod

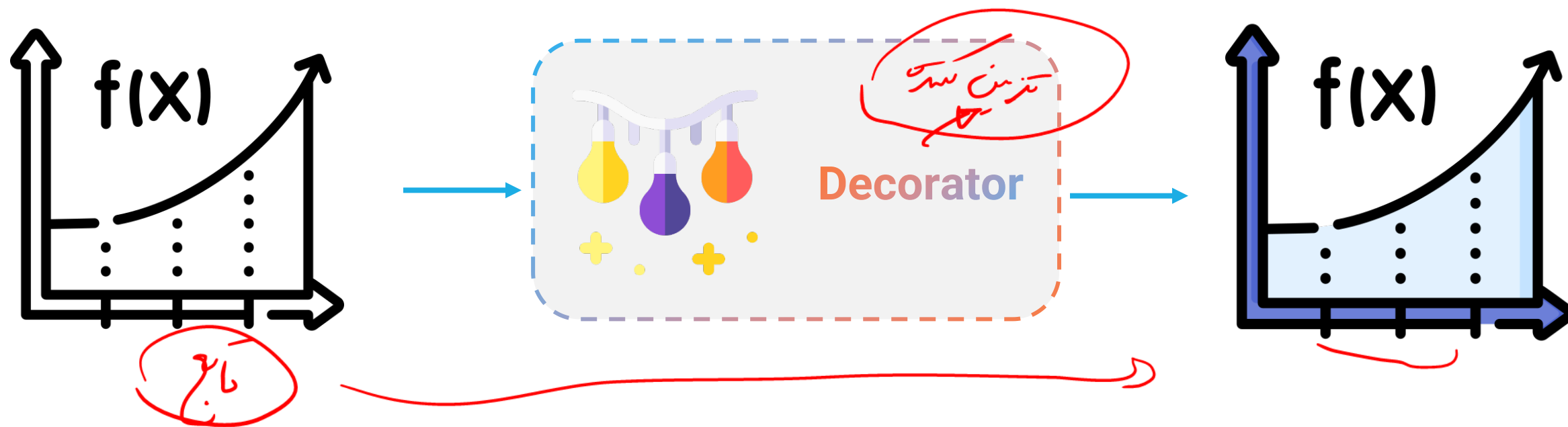
raise

@classmethod

def set_raise_parameter(cls, amount):
cls.raise_parameter = amount

Decorator

Decorator in Simple Words



Class Method As Alternative Constructor

(168, 72, h —)

```
human_1_str = "168-72-hamed-ghasemi"
```

```
human_2_str = "172-75-amin-bemani"
```

```
h, w, first, last = human_1_str.split("-")
```

```
h, w = int(h), int(w)
```

```
human_1 = Human(h, w, first, last)
```

```
human_1.introduce()
```

ایراد این روش چیست؟

h — split
o

Class Method As Alternative Constructor

—init—

```
@classmethod
def from_string(cls, input_str):
    h, w, first, last = input_str.split("-")
    h, w = int(h), int(w)
    return cls(h, w, first, last)
```

```
human_1_str = "168-72-hamed-ghasemi"
```

```
human_2_str = "172-75-amin-bemani"
```

```
human_1 = Human.from_string(human_1_str)
```

```
human_2 = Human.from_string(human_2_str)
```

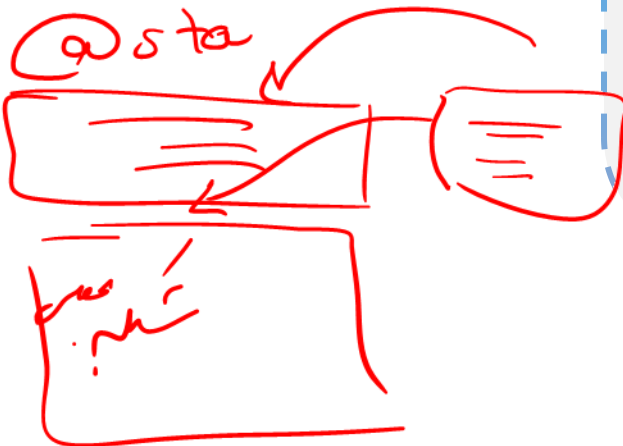
```
human_1.introduce()
```

```
human_2.introduce()
```

Static Method

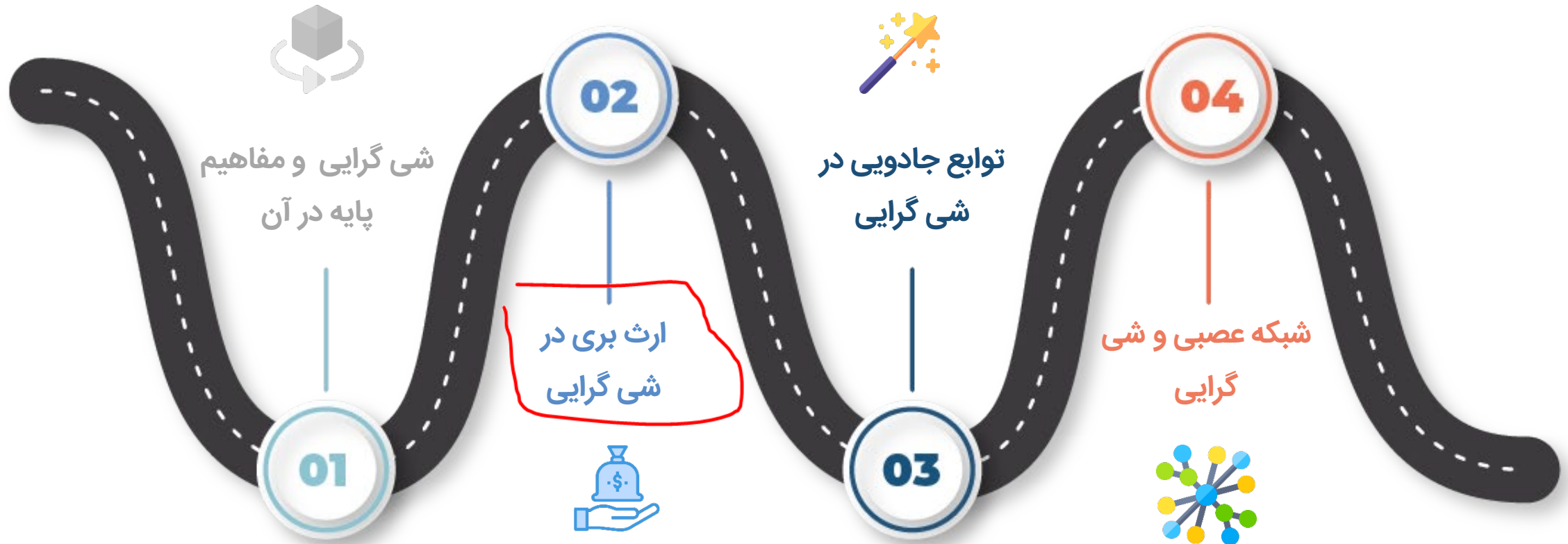
instance
class

وی به این روش



```
@staticmethod  
def creation_day():  
    print(date.today())  
  
human_1_str = "168-72-hamed-ghasemi"  
  
human_1 = Human.from_string(human_1_str)  
  
human_1.creation_day()
```

آنچه امروز خواهیم گفت :



ارث بری در شی گرای





دیگه

به کمک ارث بری می توانیم Attribute ها و Method ها از یک کلاس دیگر
را به ارث ببریم و از آنها استفاده کنیم یا آنها را تغییر دهیم بدون آن که کلاس
به ارث برده شده دچار تغییر شود.

یک مثال ساده

```
class Man(Human):  
    pass
```

```
person_1 = "168-72-hamed-ghasemi"
```

```
man_1 = Man.from_string(person_1)
```

```
print(man_1.h)  
man_1.introduce()
```


Method Resolution Order

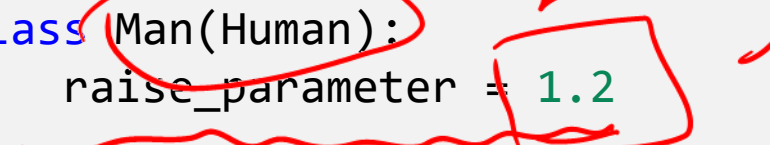
MRO

```
print(help(Man))
```



```
class Man(Human)
|   Man(height, weight, first, last)
|
|   Method resolution order:
|       Man
|       Human
|       builtins.object
|
|   Methods inherited from Human:...
```

یک مثال



```
class Man(Human):  
    raise_parameter = 1.2  
  
man_1 = Man.from_string(person_1)  
human_1 = Human.from_string(person_1)  
  
print(man_1.overweight())  
print(human_1.overweight())
```

استفاده از super

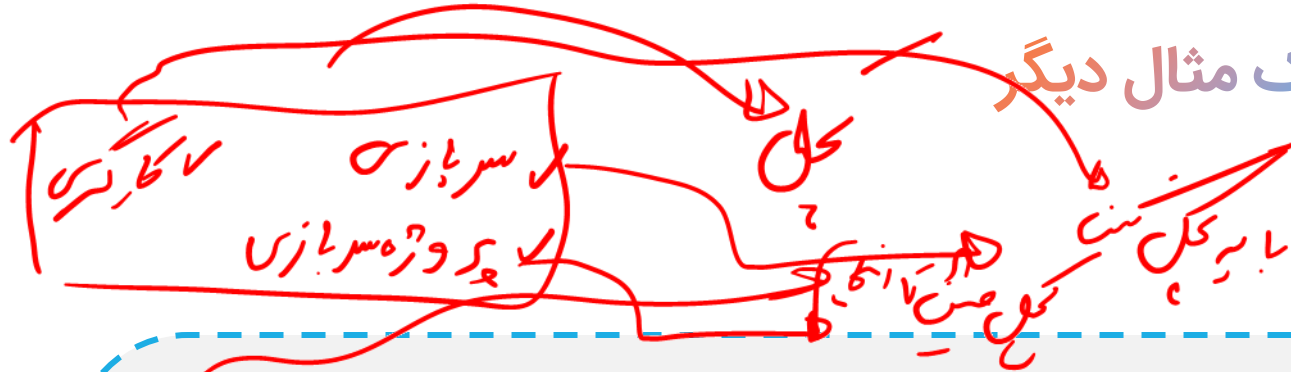
__init__ self Man *مرد*

```
def __init__(self, height, weight, first, last, marriage):  
    super().__init__(height, weight, first, last)  
    self.marriage = marriage
```

```
man_1 = Man(178, 72, "emad", "amini", 34)
```

```
print(man_1.first)
```

یک مثال دیگر



```
class Woman(Human):
```

```
    def __init__(self, height, weight, first, last, marriage, proficiency = None):
        super().__init__(height, weight, first, last)
        self.marriage = marriage
        if proficiency == None:
            self.proficiency = []
        else :
            self.proficiency = proficiency
```

کلاس

proficiency = None

[]

اضافه کردن دو متد

اضافه کردن متد

```
def add_proficiency(self, proficiency):  
    if proficiency not in self.proficiency:  
        self.proficiency.append(proficiency)  
  
def remove_proficiency(self, proficiency):  
    if proficiency in self.proficiency:  
        self.proficiency.remove(proficiency)
```

ما سی به کپی پیست می‌کنیم

↓

ما دقت
Copilot

↓

آماده

یکی دو تا تابع کاربردی

print(instance(woman_1, Woman))
print(instance(woman_1, Man))
print(issubclass(Man, Human))
print(issubclass(Man, Woman))

Instance

Class

class

sub-

یہ جمع بندی کنیم تا همین جای ماجرا رو

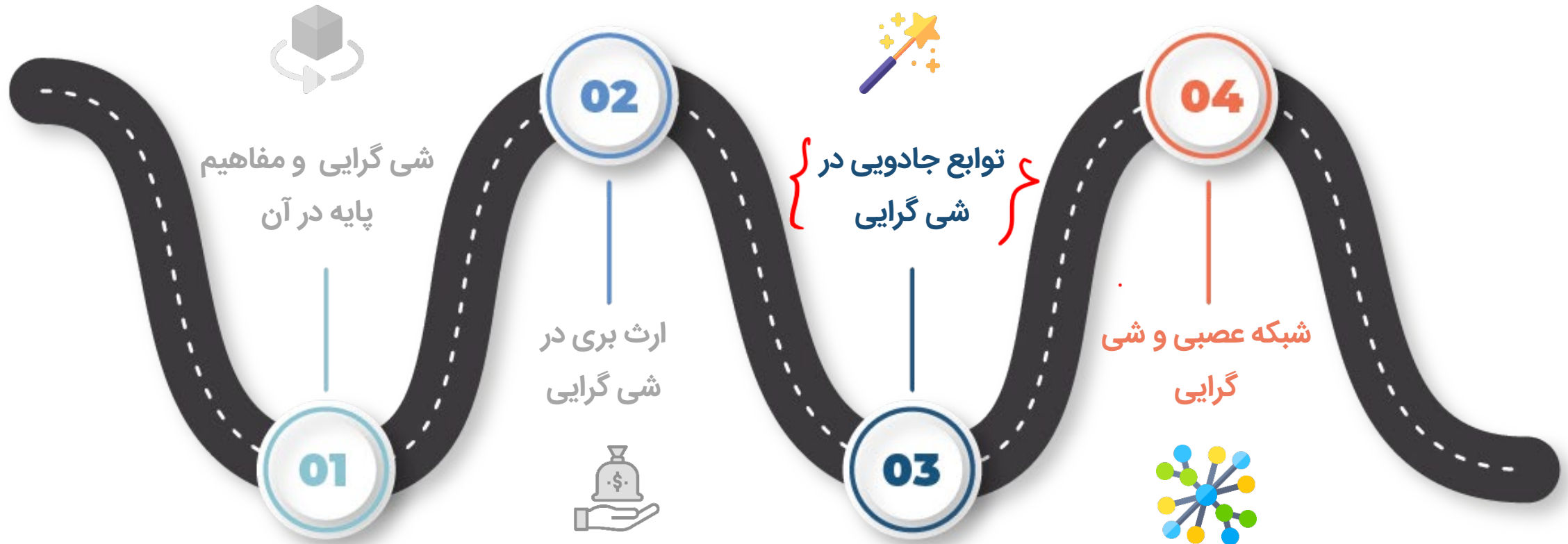
✓ init ← اس کی طرف سے خدایا بڑا ہوس
def
==

class, instance variable ✓

super ✓ اسی کی

Regular method ✓
Static Method, class method ✓

آنچه امروز خواهیم گفت :



Special Method in Python

```
print(1+2)  
print("a" + "b")
```

```
human_1 = Human(178, 71, "hamed", "ghasemi")  
print(human_1)
```



```
<__main__.Human object at 0x00000253C621DFD0>
```

متد `__repr__()`

```
def __repr__(self):  
    return "Human({}, {}, {}, {})".format(self.h, self.w, self.first, self.last)  
  
human_1 = Human(178, 71, "hamed", "ghasemi")  
  
print(human_1)
```

متد `__str__()`

```
def __str__(self):  
    return "{} {}, w = {}, h = {}".format(self.first, self.last, self.h, self.w)
```

```
print(repr(human_1))  
print(str(human_1))
```

```
print(human_1.__repr__())  
print(human_1.__str__())
```

متد `__add__()`

```
print(1 + 2)
```

```
print(int.__add__(1, 2))  
print(str.__add__("a", "b"))
```

```
def __add__(self, other):  
    return self.w + other.w
```

```
human_1 = Human(178, 71, "hamed", "ghasemi")  
human_2 = Human(182, 81, "hamed", "ghasemi")  
  
print(human_1 + human_2)
```

و البته عملگرهای دیگری نیز وجود دارد.

<i>client operation</i>	<i>special method</i>	<i>description</i>
<code>x + y</code>	<code>__add__(self, other)</code>	sum of x and y
<code>x - y</code>	<code>__sub__(self, other)</code>	difference of x and y
<code>x * y</code>	<code>__mul__(self, other)</code>	product of x and y
<code>x ** y</code>	<code>__pow__(self, other)</code>	x to the yth power
<code>x / y</code>	<code>__truediv__(self, other)</code>	quotient of x and y
<code>x // y</code>	<code>__floordiv__(self, other)</code>	floored quotient of x and y
<code>x % y</code>	<code>__mod__(self, other)</code>	remainder when dividing x by y
<code>+x</code>	<code>__pos__(self)</code>	x
<code>-x</code>	<code>__neg__(self)</code>	arithmetic negation of x

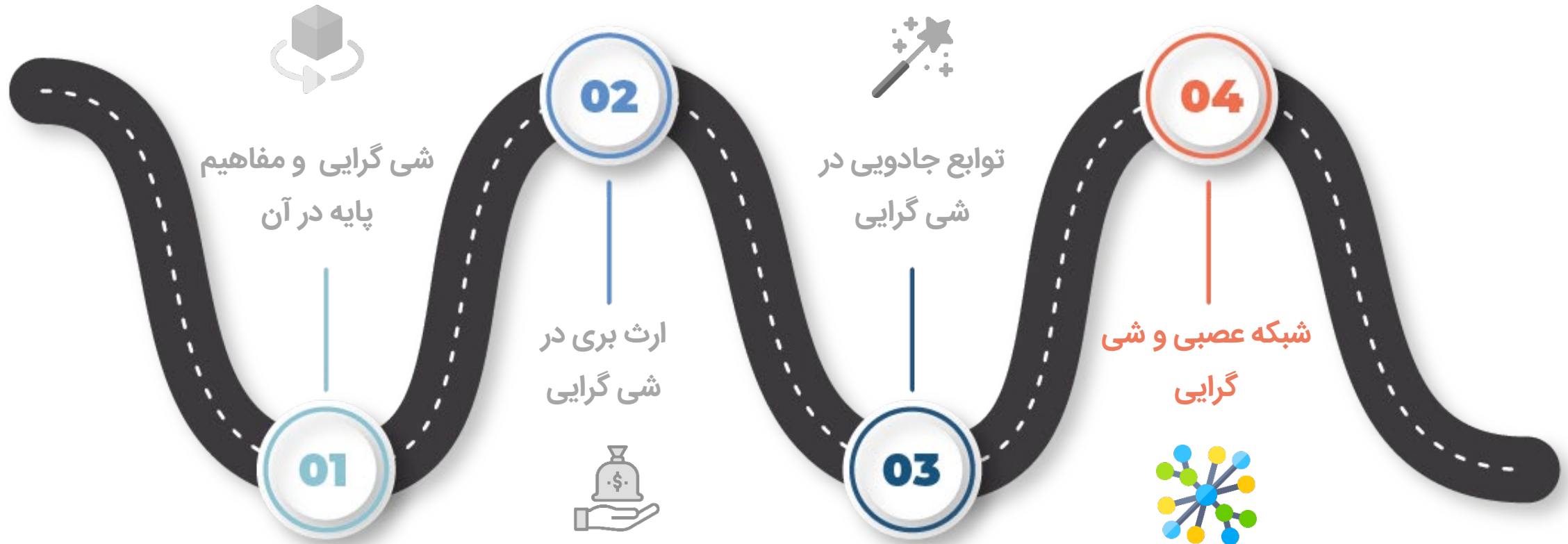
Note: Python 2 uses `__div__` instead of `__truediv__`.

Special methods for arithmetic operators

متد __len__

```
def __len__(self):  
    return len(self.first) + len(self.last)  
  
human_1 = Human(178, 71, "hamed", "ghasemi")  
  
print(len(human_1))
```

آنچه امروز خواهیم گفت :

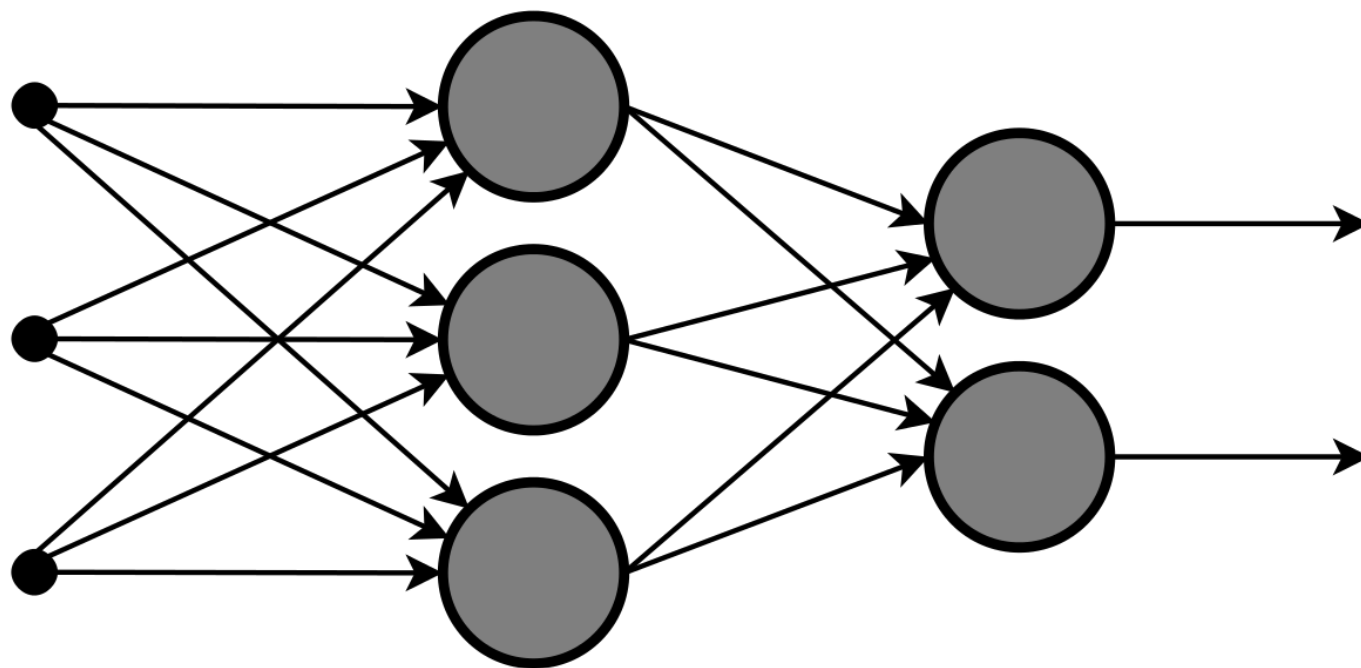


مثال یک شبکه عصبی با OOP

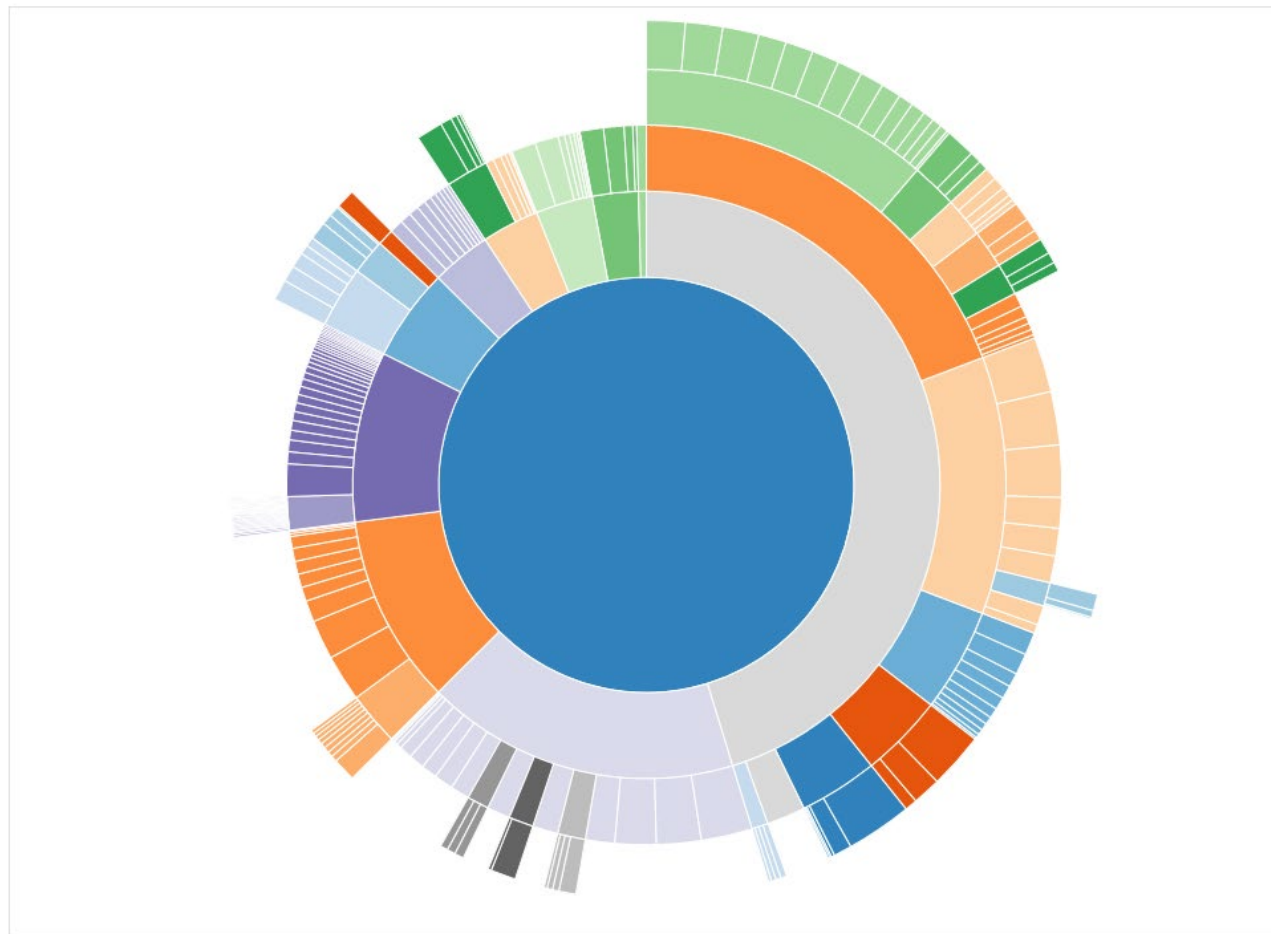
```
class MiniVGGNet():  
  
    def build(self):  
        net = models.Sequential([  
            layers.Conv2D(32, (3, 3), activation="relu", input_shape=(32, 32, 3)),  
            layers.MaxPool2D(),  
            ...  
            layers.Flatten(),  
            layers.Dense(64, activation="relu"),  
            layers.Dense(10, activation="softmax")  
        ])  
  
        return net  
  
model = MiniVGGNet()  
net = model.build()
```


می توان کمی بهینه تر کرد ؟

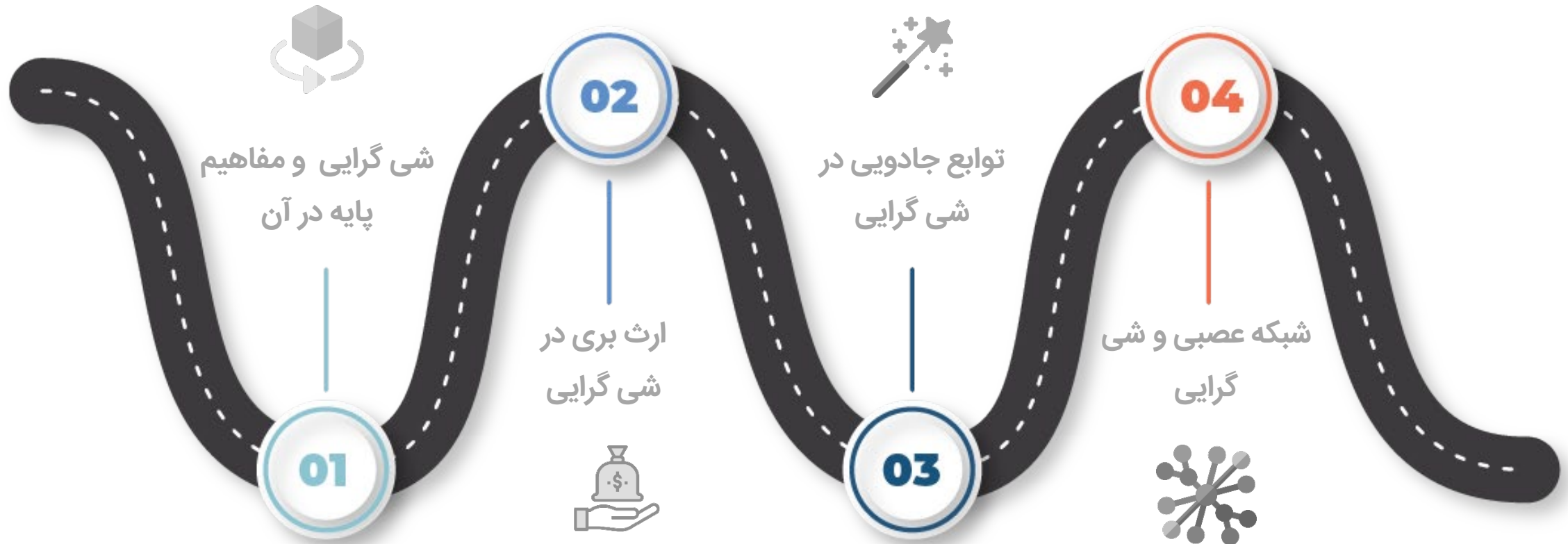
بهتر است اگر شبکه خود را در یک فایل جداگانه بنویسیم.



بهتر است اگر شبکه خود را در یک فایل جداگانه بنویسیم.



آنچه امروز خواهیم گفت :



منابع مورد استفاده



COREYMS

Development, Design, DIY, and more



coreyms.com

